Multimedia Systems
Lecture – 24

By

Dr. Priyambada Subudhi

Assistant Professor

IIIT Sri City

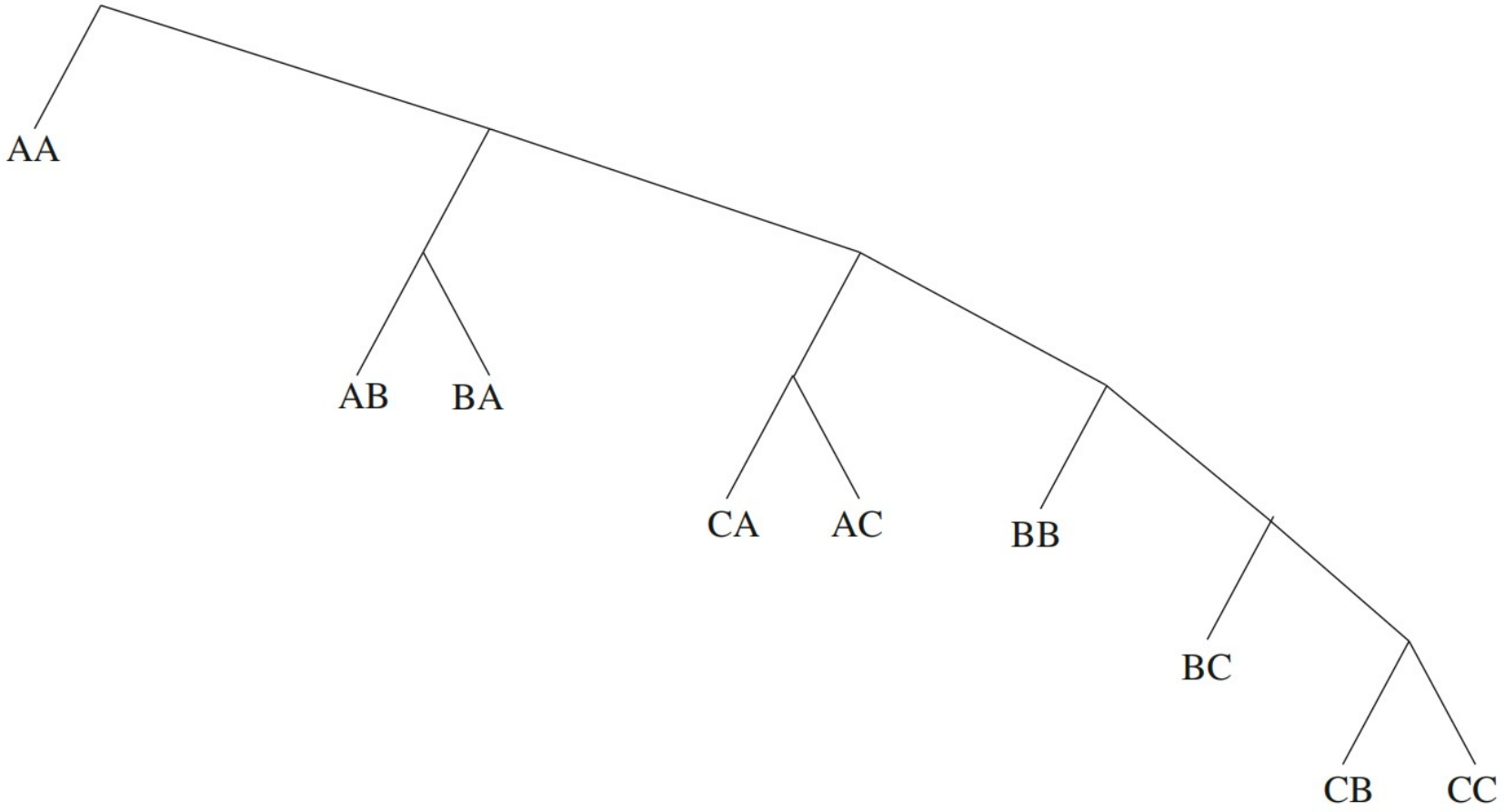
- If correct probabilities (“prior statistics”) are available and accurate, the Huffman coding method produces good compression results.
- Decoding for the Huffman coding is trivial as long as the statistics and/or coding tree are sent before the data to be compressed (in the file header, say). This overhead becomes negligible if the data file is sufficiently large.

- The following are important properties of Huffman coding:
- Unique prefix property: No Huffman code is a prefix of any other Huffman code. For instance, the code 0 assigned to L is not a prefix of the code 10 for H or 110 for E or 111 for O; nor is the code 10 for H a prefix of the code 110 for E or 111 for O.
- It turns out that the unique prefix property is guaranteed by the above Huffman algorithm, since it always places all input symbols at the leaf nodes of the Huffman tree.
- This property is essential and also makes for an efficient decoder, since it precludes any ambiguity in decoding.

- Optimality: The Huffman code is a minimum-redundancy code. It has been proven that the Huffman code is optimal for a given data model.
  - The two least frequent symbols will have the same length for their Huffman codes, differing only at the last bit.
  - Symbols that occur more frequently will have shorter Huffman codes than symbols that occur less frequently.
  - The average code length for an information source S is strictly less than $\eta + 1$. we have $\eta \leq l < \eta + 1$.

- Example: Construct a binary Huffman code for a source S with just three symbols A, B, and C, having probabilities 0.6, 0.3, and 0.1, respectively. Find out the average number of bits required and entropy.
- Now let us extend this code by grouping symbols into 2-character groups—i.e., we use blocks of k = 2 characters. We wish to compare the performance now, in bits per original source symbol.

Extended Huffman Tree



- Codeword bitlengths are then as follows:

Symbol group	Probability	Codeword	Bitlength
AA	0.36	0	1
AB	0.18	100	3
BA	0.18	101	3
CA	0.06	1100	4
AC	0.06	1101	4
BB	0.09	1110	4
BC	0.03	11110	5
CB	0.03	111110	6
CC	0.01	111111	6

- Consequently, the average bitrate per symbol is:

$$\text{Average} = 0.5\times(0.36 + 3 \times 0.18 + 3 \times 0.18 + 4 \times 0.06 + 4 \times 0.06 + 4 \times 0.09 + 5 \times 0.03 + 6 \times 0.03 + 6 \times 0.01) = 1.3350.$$

- The average bitrate per symbol was 1.4 for length-1 symbols, and the best possible is the entropy: $\eta \approx 1.2955$.
- So we found that, indeed, the Extended Huffman bitrate does fit into the bound and in fact does result in a modest improvement in compression ratio in practice.

Adaptive Huffman Coding

- The Huffman algorithm requires prior statistical knowledge about the information source, and such information is often not available.
- This is particularly true in multimedia applications, where future data is unknown before its arrival, as for example in live (or streaming) audio and video.
- Even when the statistics are available, the transmission of the symbol table could represent heavy overhead.
- For the non-extended version of Huffman coding, the above discussion assumes a so-called order-0 model—that is, symbols/characters were treated singly, without any context or history maintained.
- One possible way to include contextual information is to examine k preceding (or succeeding) symbols each time; this is known as an order-k model.
- Nevertheless, this again implies that much more statistical data has to be stored and sent for the order-k model when $k \geq 1$.

- The solution is to use adaptive compression algorithms, in which statistics are gathered and updated dynamically as the datastream arrives.
- The probabilities are no longer based on prior knowledge but on the actual data received so far.
- The new coding methods are “adaptive” because, as the probability distribution of the received symbols changes, symbols will be given new (longer or shorter) codes. This is especially desirable for multimedia data, when the content (the music or the color of the scene) and hence the statistics can change rapidly.